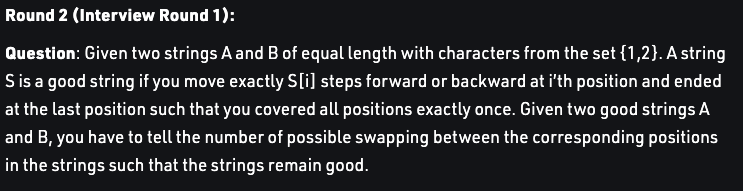
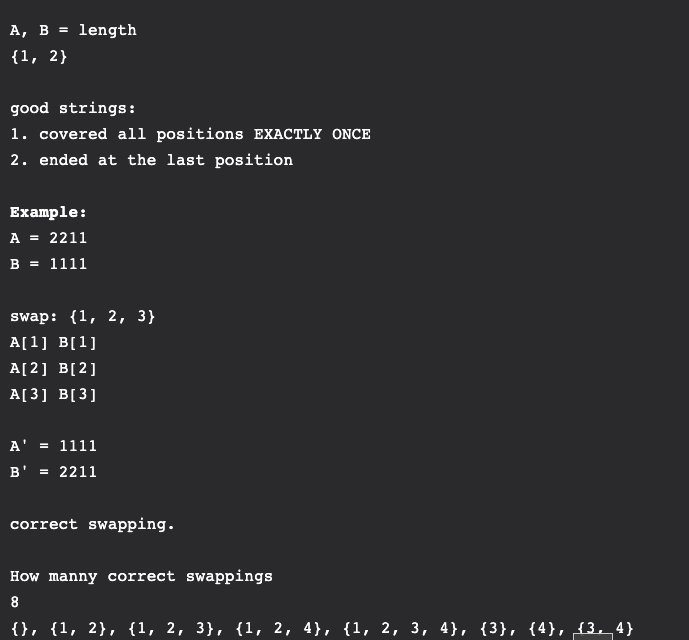
**Problem :**

****

****

**Approach :**

-> We can treat it like a normal subsequence DP problem.(**Take , not take approach**)

-> dp[i] =not taking current number,how many subsequences formed by i+1...n

taking current number,how many subsequences formed by i+1...n

Add both and return.

**=> Here there is a catch, we should be able to reach end of the array.**

-> If there is a 1, its ok

- But if we encounter a 2, the strict pattern that should follow is **221.**

**Only in this pattern 221, we are able to jump out of this pattern of 2.**

-> So we can continue the problem like a : **SWAP , NOT SWAP** approach.

But **we need 3 states dp[cur][prev1][prev2] :**

**cur = current index.**

**prev1 = 0, if there are 0 2s before this index in 1st string.**(means a 1 in previous iteration)

(in this case we don’t care whatever we put here).

**prev1 = 1, if there is one 2 before this index in 1st string.**

(we need to compulsorily put a 2 here)

**prev1 = 2, there are 2 continuous 2s before this index**(so need to compulsorily put a 1 here)

**same with prev2 for 2nd string.**

So for current index , whether we can swap or not, depends on prev1 and prev2, that is why 3 states required.

**TIme = O(n).**

**CODE :** [**https://ideone.com/SEuUiZ**](https://ideone.com/SEuUiZ)